



CENTER FOR TRUSTWORTHY
SCIENTIFIC CYBERINFRASTRUCTURE
The NSF Cybersecurity Center of Excellence

Security Log Analysis Training

Mark Krenz

*2017 NSF Cybersecurity Summit
August 15th, 2017*

<http://hdl.handle.net/2022/21717>

Center for Trustworthy Cyberinfrastructure

The NSF Cybersecurity Center of Excellence

CTSC's mission is to provide the NSF community a coherent understanding of cybersecurity's role in producing trustworthy science and the information and know-how required to achieve and maintain effective cybersecurity programs.



Speaker Bio - Mark Krenz

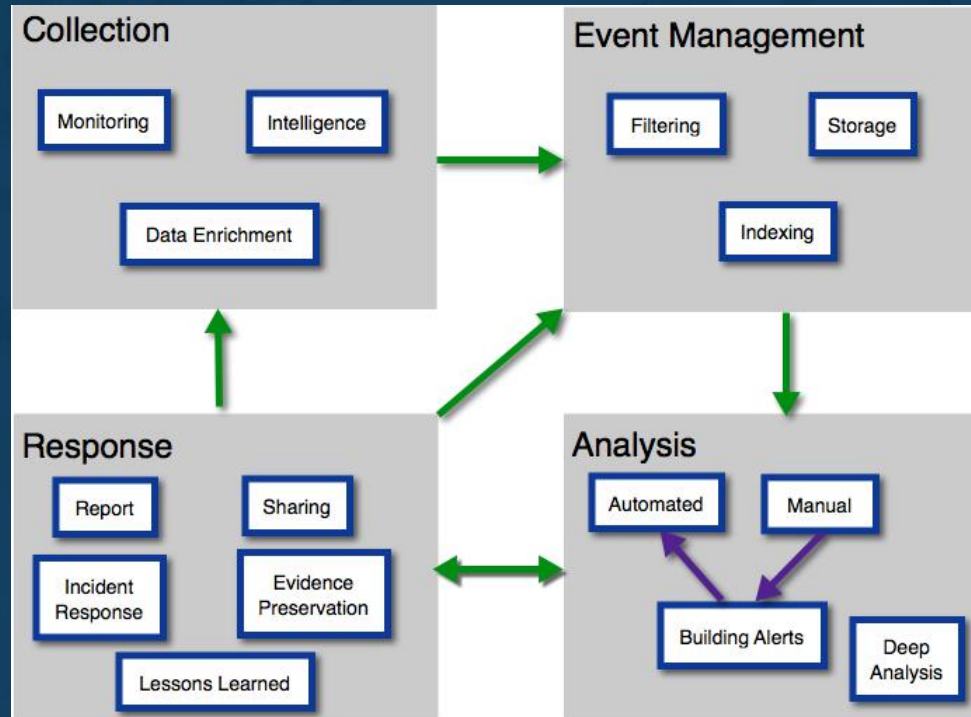
- Lead Security Analyst at Indiana University CACR (5 years)
- Part of the CTSC group
- Linux System Administrator for 20 years
- Have worked in various sectors (private, government, academic)
- Creator of popular Twitter feed @climagic that :
<https://twitter.com/climagic>

Security Log Analysis Tutorial Goals



- Today you can expect us to:
 - Take you thru the Log Analysis Lifecycle
 - Provide log analysis examples of real attacks with logs from Bro, Apache, Postfix, Duo, OpenSSH, etc.
 - Encourage interactive Q&A throughout
- You should take away
 - Ideas to improve your security logging & monitoring
 - Command examples that you can customize for use on your own logs.
 - Methods you can generalize to explore and connect events across logs

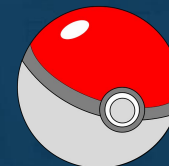
Log analysis workflow



Overview



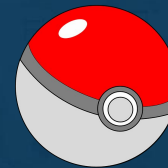
- ***Collection***
 - Log sources
 - Intelligence
 - Data Enrichment
- Event Management
- Analysis
- Response



Collection Sources



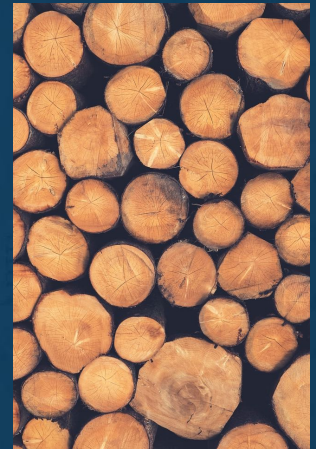
- Logs
 - System: Syslogs, Windows events, Host-based IDS, etc.
 - Network: netflows, Bro, Snort, etc.
 - Application: Apache, VPN, OAuth, etc.
- Intelligence
 - REN-ISAC SES, Critical Stack, Team Cymru MHR, ...
- Data Enrichment
 - Additional information to complement logs
 - LDAP, DHCP, Inventories



Collection - Logs



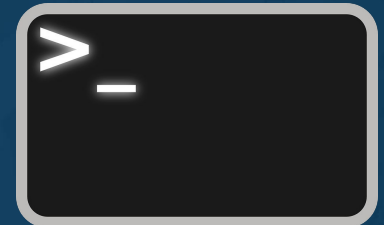
- Syslog
- Windows Event Logs
- HIDS
 - OSSEC
 - Auditd
 - Keystroke logs
- Netflows
- Bro/NIDS
- Web servers, mod_security
- DNS
- Scans



Collection - Logs - Syslog



- De Facto framework on Linux systems
- System & any software can log to it
- Can log centrally; default unreliable UDP
- Components
 - Priority
 - Timestamp
 - Hostname
 - Process
 - Message



Collection - Logs - Syslog



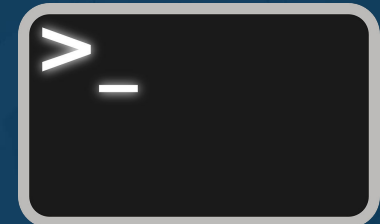
Priority (Facility + Severity): e.g. <38>, <86>

FACILITY

0	kern	kernel messages
1	user	user-level messages
2	mail	mail system
3	daemon	system daemons
4	auth	security/authorization messages
5	syslog	messages generated internally by syslogd
6	lpr	line printer subsystem
7	news	network news subsystem
8	uucp	UUCP subsystem
9		clock daemon
10	authpriv	security/authorization messages
11	ftp	FTP daemon
12	-	NTP subsystem
13	-	log audit
14	-	log alert
15	cron	scheduling daemon
16	local0	local use 0 (local0)
17	local1	local use 1 (local1)
18	local2	local use 2 (local2)
19	local3	local use 3 (local3)
20	local4	local use 4 (local4)
21	local5	local use 5 (local5)
22	local6	local use 6 (local6)
23	local7	local use 7 (local7)

SEVERITY

0	emerg	System is unusable
1	alert	
2	crit	Critical conditions
3	err	Error conditions
4	warning	
5	notice	Events that are unusual, but not error conditions.
6	info	Normal operational messages that require no action.
7	debug	Information useful to developers for debugging the application.



Syslog Examples - SSH



<38>Aug 1 09:13:58 groot sshd[19468]: Accepted publickey for wraquel from 10.12.23.15 port 49474 ssh2: RSA 2b:cb:82:f0:22:d7:8a:f6:cd:70:43:b3:de:cf:5d:ee

<86>2016-08-01T09:13:48.764820-05:00 bastion sshd[2193]: Accepted keyboard-interactive/pam for wraquel from 10.12.23.15 port 49458 ssh2

<38>Aug 1 14:05:17 dev2 sshd[31622]: Failed password for root from 10.11.128.16 port 48593 ssh2

<38>Aug 1 09:37:20 honeypot sshd[9256]: Failed password for invalid user pi from 192.168.58.61 port 59699 ssh2

Syslog issues

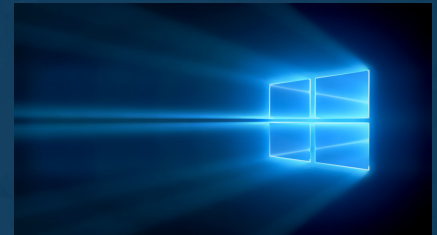


- Only so much can be conveyed in text
- Unicode can sometimes cause issues
- Relays may include additional relay hostnames
- Timestamps may be off
- Timezone discrepancies

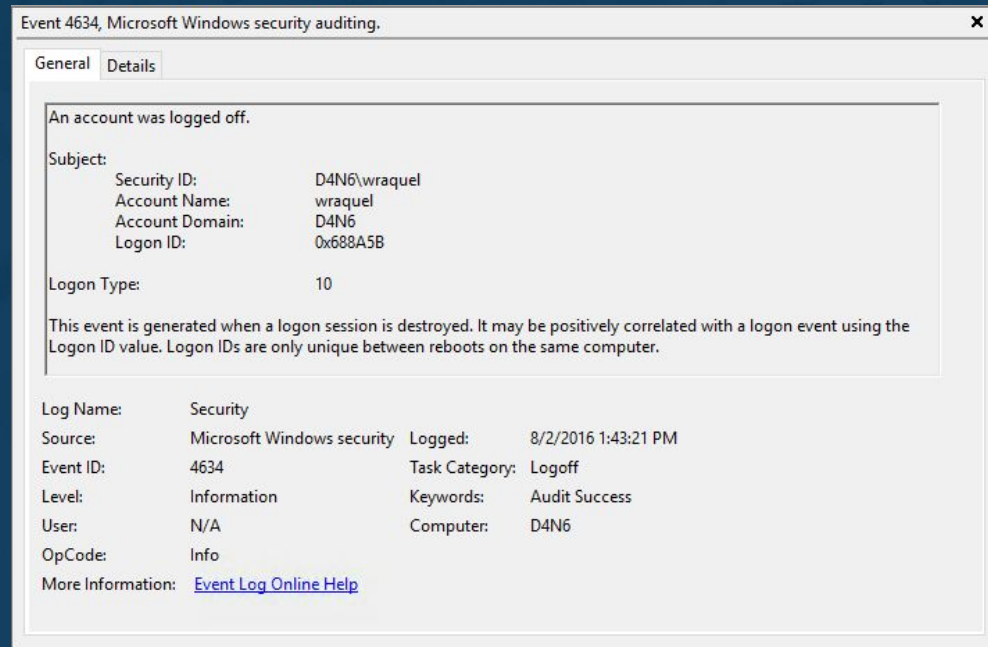
Collection - Logs - Windows Event Logs



- Stored locally; Overwrites itself
- Can be centrally collected
- Unicode
- Very Verbose
- Central Storage a.k.a. Subscriptions



Collection - Logs - Windows Event Logs



Collection - Logs - Windows Event Logs



Event 4634, Microsoft Windows security auditing.

General Details

☐ Friendly View ☒ XML View

```
- <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
- <System>
  <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54849625-
    5478-4994-A5BA-3E3B0328C30D}" />
  <EventID>4634</EventID>
  <Version>0</Version>
  <Level>0</Level>
  <Task>12545</Task>
  <Opcode>0</Opcode>
  <Keywords>0x8020000000000000</Keywords>
  <TimeCreated SystemTime="2016-08-02T18:43:21.906378400Z" />
  <EventRecordID>13558</EventRecordID>
  <Correlation />
  <Execution ProcessID="1048" ThreadID="8144" />
  <Channel>Security</Channel>
  <Computer>D4N6</Computer>
  <Security />
</System>
- <EventData>
```

Collection - Logs - Windows Event Logs

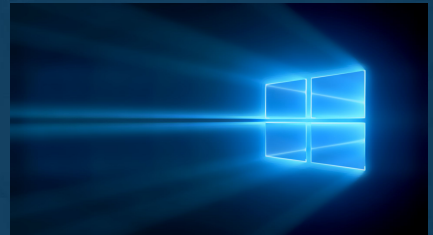


Aug 2 13:43:21 D4N6 MSWinEventLog#0111#011Security#0 Aug 02 13:43:21
2016#0114634#011Microsoft-Windows-Security-Auditing#011N/A#011N/A#011Success Audit#011D4N6#011Logoff#011#011An account was logged off. Subject: Security ID: S-1-5-21-3934507682-2419030825-887421081-1004 Account Name: wraquel Account Domain: D4N6 Logon ID: 0x688A5B Logon Type: 10 This event is generated when a logon session is destroyed. It may be positively correlated with a logon event using the Logon ID value. Logon IDs are only unique between reboots on the same computer.#01113558#015

Collection - Logs - Windows Event Logs



- Working in mixed environments
 - Subscription service to collect centrally
 - Can be converted to Syslog
 - Can be sent directly to some information managers



Collection - Logs - HIDS



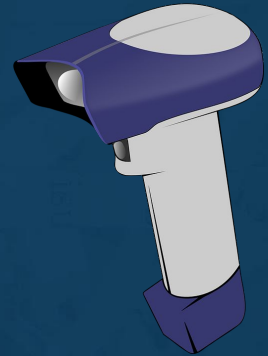
- Usually provides
 - File Integrity
 - Rootkit detection
 - Log Monitoring
 - Process Monitoring



Collection – Scan Logs



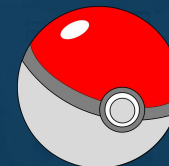
- Sources
 - NMAP
 - Qualys
 - Masscan



Overview



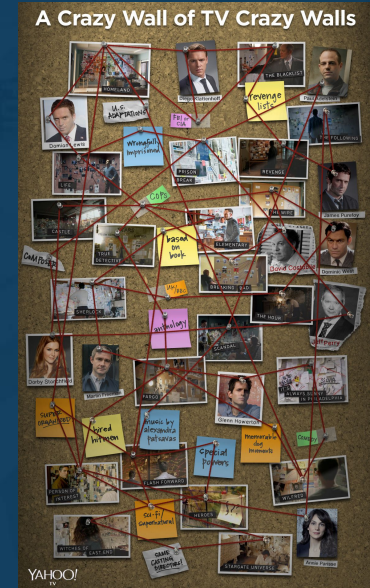
- Collection
 - Log sources
 - ***Intelligence***
 - Data Enrichment
- Event Management
- Analysis
- Response





Collection - Intelligence

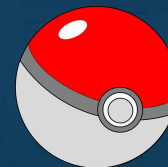
- Collective Intelligence Framework (CIF)
- MISP
- OSINT (Open Source Intelligence)
- Critical Stack
- whitelisting/blacklisting
- Internal business information



Overview



- Collection
 - Log sources
 - Intelligence
 - ***Data Enrichment***
- Event Management
- Analysis
- Response



Collection - Data Enrichment



- Helps define non-descript data
- Types
 - hostnames
 - groups
 - users
- Sources



nInfo example



<https://github.com/JustinAzoff/ninfo>

```
$ ninfo -p geoip -p cymruwhois 8.8.8.8 4.2.2.2
=== 8.8.8.8 ===
*** Cymru Whois (Cymru Whois lookup) ***
15169 US 8.8.8.0/24 GOOGLE - Google Inc.

*** GeoIP (GeoIP) ***
US - United States

=== 4.2.2.2 ===
*** Cymru Whois (Cymru Whois lookup) ***
3356 US 4.0.0.0/9 LEVEL3 Level 3 Communications

*** GeoIP (GeoIP) ***
US - United States
```


Collection - Data Enrichment



- Helps define non-descript data
- Types
- Sources
 - DHCP
 - Lifecycle Management Tools (Foreman, puppet)
 - CMDB
 - Inventory
 - Business information



Collection - Authorization



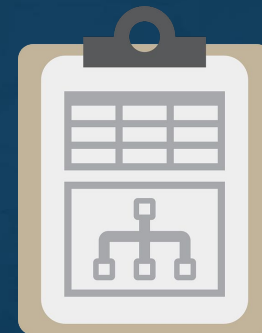
- Legal Issues
- Buy In
- Who owns the data?



Overview



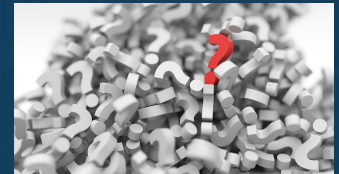
- Collection
- ***Event Management***
 - Filtering
 - Storage
 - Indexing
 - Accessibility
- Analysis
- Response



IEM – Considerations



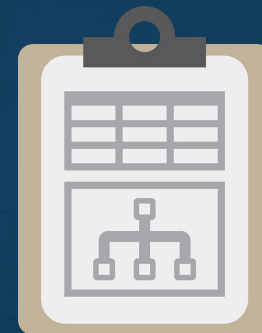
- How do we send logs to a central location?
- What format are they in?
- Can this format be converted?
- How will we store this information for later retrieval?
- What rate is all this data coming in at?
- Do I want to store all this data or some of it?



Overview



- Collection
- Event Management
 - ***Filtering***
 - Storage
 - Indexing
 - Accessibility
- Analysis
- Response



Filtering

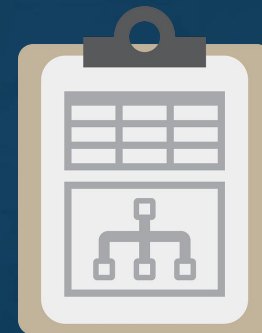


- Reducing amount of incoming data
 - Can we drop useless messages?
 - Do we need all these sources?
- Sorting on the fly
 - Timestamps
 - Message Types
 - Categorization

Overview



- Collection
- Event Management
 - Filtering
 - ***Storage***
 - Indexing
 - Accessibility
- Analysis
- Response



Storage Goals – Information and Event Management



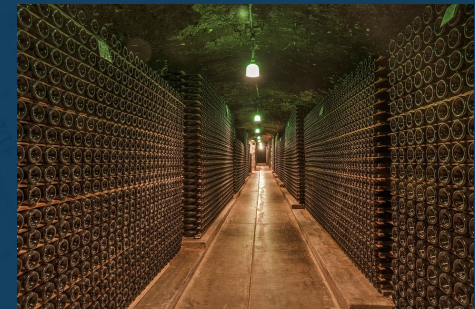
- Centralized access to information
- Consistent format
- Ease of access
 - Machine parseable
 - Human Readable
 - Categorized





IEM – Storage Concerns

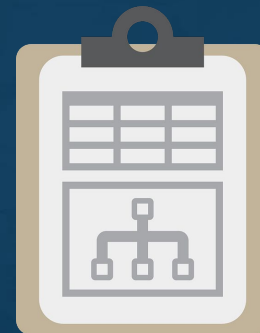
- Scalability
- I/O Limitations
- Long Term storage
 - FOIA Requests
 - Archivists
- Categorization
- Timestamp issues
- Raw vs Parsed



Overview



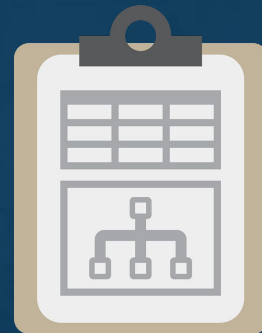
- Collection
- Event Management
 - Filtering
 - Storage
 - ***Indexing***
 - Accessibility
- Analysis
- Response



Indexing



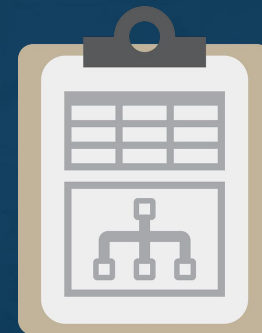
- Quicker access to raw data
- Keywords
- Timestamps/Frames
- Summaries



Overview



- Collection
- Event Management
 - Filtering
 - Storage
 - Indexing
 - ***Accessibility***
- Analysis
- Response



IEM - Accessibility & Interfaces



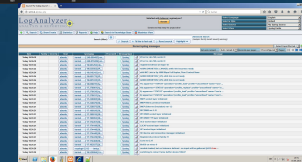
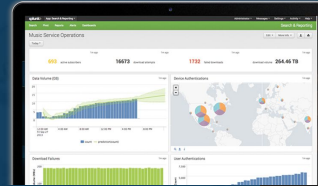
- Dashboard Access
- Direct log access
- Permissions
- Data Sensitivity



IEM – Log Management Tools



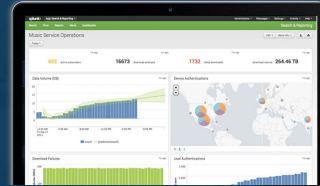
- Splunk
- ELK



IEM – Log Management Tools



- May take years to tune
- Specialized Training
- Black box



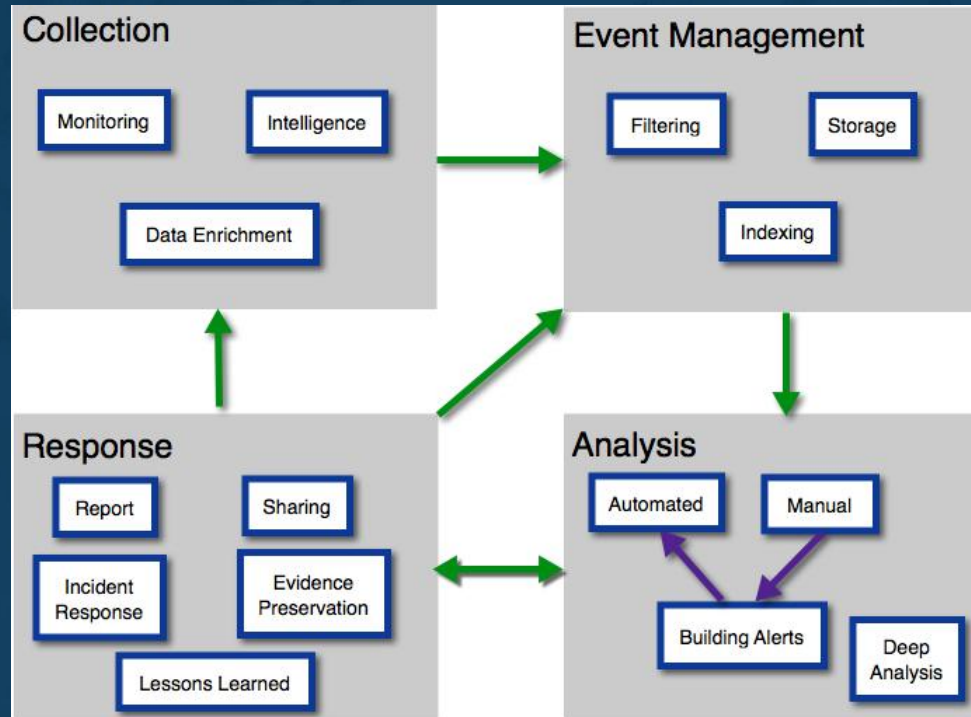
Watch out: Useless metrics



- Don't be fooled by useless graphics and charts



Log analysis workflow



Overview



- Collection
- Event Management
- ***Analysis***
- Response



Analysis



- Manual
- Alerts
- Automated
- Deep Dives



Analysis



- ***Manual***
 - Requires domain knowledge
 - Tool proficiency
- Alerts
- Automated
- Deep Dives



Analysis



- Manual
- ***Alerts***
 - Creates reports
 - Can act (block, notify)
 - Lots of tweaking
- Automated
- Deep Dives



Analysis



- Manual
- Alerts
- ***Automated***
 - Watch for false positives
 - Track actions
 - Review often
- Deep Dives



Analysis



- Manual
- Alerts
- Automated
- ***Deep Dives***
 - One off analysis
 - May require access to data you don't have access to



Analysis



- What about experience?
- Just act like a scientist.



Analysis - Scientific Method



- Make an observation
- Ask questions
- Develop testable predictions
- Test, test, test (refine)
- Develop a conclusion (was I correct?)



Introduction to Bro Logs



"This is a better kind of bro."

- Karen Sandler (Software Freedom Conservancy)

What is Bro?



- "Network Security Monitor"
- Inspects all network traffic
- Generates forensically sound logs
 - See "Network Forensics With Bro" by Matthias Vallentin
- Has a scripting language, making it very extensible
 - automation, custom functionality, etc.
- Crucial difference:
 - IDS is logging "bad" things
 - NSM is logging everything
- What if you didn't know that something was bad?

Bro Logs



- Goal: Determine what happened on the network
- Logs are small
- Logs are plain-text, tab-delimited CSV
- Designed for grep, sed, awk, sort, head, etc.

Bro Log Limitations



- Logs are an abstraction
 - 100s of Gbps of input, a few GB/hour of output requires loss of data
- Only some protocols can be parsed
- Rising use of encryption
- Limited by network visibility

Example 1: IRC



141.142.11.2:4766 -> 164.32.77.23:6667

JOIN #foobar

irc.log	Value
id.orig_h	141.142.11.2
id.orig_p	4766
id.resp_h	164.32.77.23
id.resp_p	6667
nick	USA 74634
user	[urX]-700159
command	JOIN
v	
value	#foobar

Example 2: HTTP



> GET /phpBB HTTP/1.1

< HTTP/1.1 404 Not Found

http.log	Value
id.orig_h	192.168.1.20
id.orig_p	7182
id.resp_h	141.142.192.147
id.resp_p	80
method	GET
host	www.ncsa.edu
uri	/phpBB
v	
status_code	404

Types of Bro Logs: Network Protocols



- conn
- dhcp
- dns
- ftp
- http
- irc
- kerberos
- mysql
- radius
- rdp
- sip
- smtp
- snmp
- socks
- ssh
- ssl
- syslog
- tunnel

Types of Bro Logs: Files



- **files.log**
- **pe.log** (Portable Executable)
- **x509.log** (Certificate information)

Types of Bro Logs: Detection



- **intel.log** (Intelligence data hits)
- **notice.log** (Alerts from Bro scripts)
- **signatures.log** (Traffic signature hits)

Types of Bro Logs: Network Observations



- **known_certs.log** (SSL certs observed)
- **known_devices.log** (MAC addresses)
- **known_hosts.log** (IPs w/ established TCP)
- **known_services.log** (Open ports)
- **software.log** (Software w/ version info)

Bro - Viewing Logs



- less
- grep
- bro-cut
- try.bro.org
- ELK Stack / Elastic Stack
- Splunk

Bro - Viewing Logs (more commands)



- `cat` and `zcat`
- `awk`
- `sort`
- `uniq`
- `wc`
- `head` and `tail`
- while `read -r MyVar` ;
do `something MyVar` ;
done

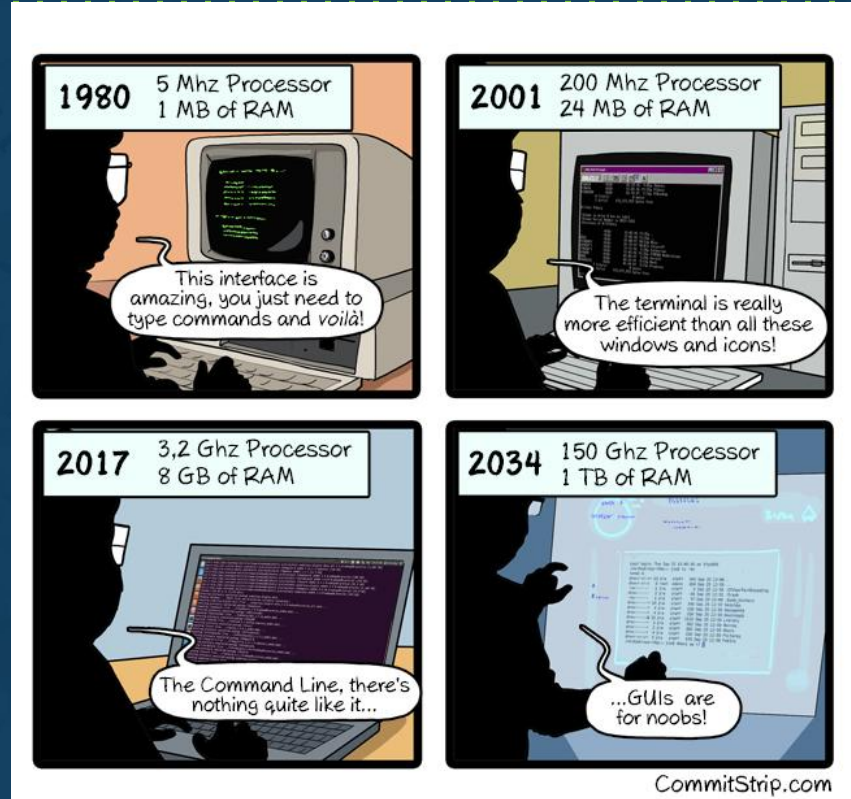


Image source: <http://www.commitstrip.com/en/2016/12/22/terminal-forever/>

Command syntax (grep)



Pattern:

```
grep [options] <search pattern> [file1] [file2] [...]
```

Real examples:

```
grep -i administrator syslog
```

```
grep -i administrator syslog | grep -v 172\.\16\.\0\.
```

Command syntax (awk)



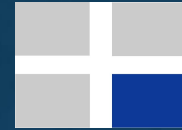
Pattern:

awk [options] <'program'> [file1] [file2] [...]

Starter program keywords:

- {print \$0} (action statements)
- \$1, \$2, ..., \$NF
- \$2=="foo", \$2!="foo"
- \$3~/^[Bb]etty\$/
- true || false, true && true
- ()
- variable=value

How a command pipeline works



- Read in data, send output to next command
- Example (search for IP and count instances)

```
$ grep 10\.100\.52\.5 conn.log | wc -l  
→ 507
```

- Example (show list of IPs ordered by count)

```
$ zcat conn.log.gz | awk -F\t '{print $3}' | sort | uniq -c | sort -rn  
→ 155489 172.16.0.10  
2836 172.16.0.5  
1456 172.16.0.13  
813 172.16.0.2  
64 172.16.0.7
```




Brief regular expression primer

- A regex can be used to match patterns of text data.
- Use " " or ' ' to protect expression from shell interpretation.
- `.` – matches any single character
- `\.` – Matches a literal `.` (use a `\` before any special character)
- `.*` – matches any character zero or more times
- `.+` – matches any character 1 or more times
- `^` – Matches the beginning of the line.
- `$` – Matches the end of the line
- `[a-z]` – matches any letter between a and z in 1 position
- `[a-zA-Z0-9]` – Matches any alphanumeric in ASCII
- `[^0-9]` – Matches any character that is not 0 through 9.
- `[0-9]{1,3}` – Matches any character 0 - 9 between 1 and 3 times
- `(apple|banana|cherry)` – Matches any of these words

Regex precision is important

Use `^2\.4\.150\.1$` to search for the IP `2.4.150.1`

Why shouldn't I just run this?

```
grep "2.4.150.1" access_log
```

Because it will also match these IPs:

`22.4.150.15`

`204.150.100.10`

and these values:

`2E4150A1`

`/script.php?id=12948150218`

Finding Compromised Systems



Phases of Compromise



1. Reconnaissance
2. Exploitation
3. Reinforcement
4. Consolidation
5. Pillage

Phases of Compromise



1. **Reconnaissance**
2. Exploitation
3. Reinforcement
4. Consolidation
5. Pillage

Recon: Searching for exploitable code



Which IP had the most HTTP 404 Not Found errors?

- What is a 404 not found error?
 - HTTP status return code to the client
- What logs track this information?
 - Bro's http.log
- What field is it in the bro log?
 - status_code
- How can we match a number in a log? *
 - awk, grep, sed
- How can we generate a top list? *
 - Collect like groups (**sort**)
 - Count the number of items in each group (**uniq -c**)
 - Order the counts. (**sort -n**)

Recon detection command (404s)



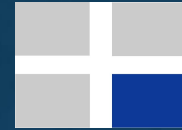
```
$ cat http.log |  
bro-cut id.orig_h status_code |  
awk -F\\t '$2=="404"' |  
sort | uniq -c | sort -n |  
tail -n 1
```

→ 165 64.39.106.131 404

```
$ dig +short -x 64.39.106.131
```

→ sn031.s01.sea01.qualys.com

Recon: Can web app read filesystem?



Do any successful queries to Wordpress code contain filesystem paths in the query string?

- Where do wordpress requests get logged?
 - access_log (on Apache)
- What should I search for?
 - Filesystem path indicators like '/', '..', '/etc' or
 - Specific filenames like my.cnf, passwd, .htaccess
- How can I figure out if the exploit attempt worked?
 - HTTP return status (if 404, then probably not; 200 only means potentially)
 - Does the file referenced exist?

Recon: Did a new exploit hit us in past?



Given that the recent Intel AMT vulnerability has been hidden in chips since 2010, can we find any indication of previous attacks against our network?

- What are we looking for?
 - meta data about traffic to tcp ports 16992 and 16993
- Where can we find this?
 - Bro's 'conn.log'
- How can we be sure the connections were successful?
 - Check that the `conn_status` column in `conn.log` is not "50".
- Make a list of potential attackers first, save it to a file.
- Then investigate the overall activity of the potentials.

Recon detection command (Intel AMT)



```
$ zcat 201[0-7]*/conn.*.log.gz |  
cat - current/conn.log |  
awk -F\\t '($6==16992 || $6==16993) && $12!="S0"{print}' |  
less -S
```



Recon detection command (Intel AMT)



```
$ zcat 201[0-7]*/conn.*.log.gz |  
  cat - current/conn.log |  
  awk -F\\t '($6==16992 || $6==16993) && $12!="S0"{print  
$3}' > potential-attackers.txt  
  
$ zgrep -F -f potential-attackers.txt  
201[0-7]*/conn.*.log.gz current/conn.log
```

Phases of Compromise



1. Reconnaissance
2. **Exploitation**
3. Reinforcement
4. Consolidation
5. Pillage

Compromise 1: http.log



id.resp_h	91.134.161.42
id.resp_p	80
method	GET
host	top4download.org
uri	/
referrer	-
user_agent	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
status_code	200
resp_mime_types	text/html

Compromise 1: http.log



id.resp_h	91.134.161.60
id.resp_p	80
method	GET
host	roseindia.vip
uri	/?644v0o1fsfarflf06=24&4e0flef...
referrer	http://top4download.org/
user_agent	(IE 11)
status_code	200
resp_mime_types	text/html

Compromise 1: http.log



id.resp_h	91.134.161.60
id.resp_p	80
method	GET
host	roseindia.vip
uri	/?644v0o1fsfarflf06=24&4e0flef...
referrer	http://top4download.org/
user_agent	(IE 11)
status_code	200
resp_mime_types	text/html

Referrer Chain



1. > GET top4download.org/ (IE 11) <
text/html
2. > GET
roseindia.vip/?644v0o1fsfarflf06=24&4e0fle186v43re1bv=1280&37fj4d7g9
4969r=720 (IE 11) <
text/html
3. > GET 18a43zad864bo96.armlay.gdn/ (IE 11) <
text/html
4. > GET 18a43zad864bo96.armlay.gdn/712g40rdf7k06 (IE 11)
< application/x-shockwave-flash

Overview for IP



91.134.161.42	GET	top4download.org	text/html
91.134.161.60	GET	roseindia.vip	text/html
62.138.5.199	GET	18a43z96.armlay.gdn	text/html
62.138.5.199	GET	18a43z96.armlay.gdn	application/x-shockwave-flash
62.138.5.199	GET	18a43z96.armlay.gdn	text/html
62.138.5.199	GET	18a43z96.armlay.gdn	application/x-shockwave-flash
62.138.5.199	GET	62.138.5.199	-
62.138.5.199	GET	62.138.5.199	application/x-dosexec
62.138.5.199	GET	62.138.5.199	application/x-dosexec
52.3.78.30	GET	ipinfo.io	text/json

Security Log Analysis

NSF Cybersecurity Summit 2017 -- August 15th, 2017



Quick Overview

Static Analysis

Behavioral Analysis

Network Analysis

Dropped Files

Comment Board (0)

Download PCAP

Domains (1)

Hosts (2049)

HTTP (1)

IRC (0)

SMTP (0)

HTTP Requests

URI	DATA
http://ipinfo.io/json	GET /json HTTP/1.1 Host: ipinfo.io

Overview for IP



1. Redirect chain with random-looking domain names, and suspicious TLDs (.vip, .gdn)
2. Shockwave Flash followed by Windows executable download
3. Queried for IP address information

Compromise 2: http.log

```
Jun 17 23:00:10 CcMeer3amA5aZ9nrx 107.160.46.226 4908 141.142.234.27 2375 1 GET 141.142.234.27 ☐
/version - - ☐
0 145 200 OK - - - (empty) - - - - - Fr5LXVyNQ3lRrs2tg text/json

Jun 18 02:10:21 CFVSv31q8HACwAJSOc 107.160.46.226 4534 141.142.234.27 2375 1 GET 141.142.234.27 ☐
/v1.23/containers/json?all=0&limit=-1&trunc_cmd=0&size=0 - python-requests/2.10.0 ☐
0 36000 200 OK - - - (empty) - - - - - Fay4vxEzVjage6cy1 text/json

Jun 18 02:10:21 CQMaBW2KP1XCGMVNlb 107.160.46.226 4533 141.142.234.27 2375 1 GET 141.142.234.27 ☐
/version - Python-urllib/2.7 ☐
0 145 200 OK - - - (empty) - - - - - FUpmS027Pvsmk0k5n4 text/json

Jun 18 02:34:35 CqA2Xg3qh9Lrpi6IEj 107.160.46.226 2516 141.142.234.27 2375 1 GET 141.142.234.27 ☐
/version - Python-urllib/2.7 ☐
0 145 200 OK - - - (empty) - - - - - FHqbUe1aylw905YFP8 text/json

Jun 18 02:34:35 CTAMVF3Rv4jhcgBRAc 107.160.46.226 2517 141.142.234.27 2375 1 POST 141.142.234.27 ☐
/v1.23/containers/6df61c916b1aee2d72046ce92bbbc16dd01c9dfb847faa12286c9e3bcd5d745c/exec - python-requests/2.10.0 ☐
216 74 201 Created - - - (empty) - - - Fds3MstwaFnM6XAw8 text/json FpxUE944g6vBSuAfkH text/json

Jun 18 02:34:35 CTAMVF3Rv4jhcgBRAc 107.160.46.226 2517 141.142.234.27 2375 2 POST 141.142.234.27 ☐
/v1.23/exec/182881b4e9e685453e610021892788085ab814518bde903c957cfdc272066d01/start - python-requests/2.10.0 ☐
31 119 200 OK - - - (empty) - - - FWK4NW22KWWiB462p1 text/json FzCk3uWDE3YjVKkb -

Jun 18 02:35:02 CaBfuW2tjnMVk7FnIl 107.160.46.226 3747 141.142.234.27 2375 1 GET 141.142.234.27 ☐
/version - Python-urllib/2.7 ☐
0 145 200 OK - - - (empty) - - - - - FISSYk4kMV0J8A9wv1 text/json

Jun 18 02:35:02 CSI7QrHUkubbD8nU1 107.160.46.226 3750 141.142.234.27 2375 1 POST 141.142.234.27 ☐
/v1.23/containers/6df61c916b1aee2d72046ce92bbbc16dd01c9dfb847faa12286c9e3bcd5d745c/exec - python-requests/2.10.0 ☐
246 74 201 Created - - - (empty) - - - FLzVNf1jnhEtYjki2j text/json FfkBeY1jz0SEpgK0K text/json
```

Exploit: Logins vs. non-work time



Can we analyze a log to show entries of login activity outside of normal working hours?

- What service do we want to check against?
 - ssh
- What logs provide this information?
 - /var/log/secure, /var/log/messages or /var/log/syslog
 - Bro's ssh.log
- How to compare time of day?
 - Use bro-cut to convert ts column to parsable local time.
 - Use awk's substr() function for hour of the day.

Break it down



```
cat ssh.log |  
bro-cut -C -d ts id.orig_h id.resp_h auth_success direction
```

[snipped]

```
#open 2017-06-24-00-00-38
```

```
#fields ts id.orig_h id.resp_h auth_success direction
```

```
#types string addr addr bool enum
```

2017-01-24T03:33:58-0400	121.48.18.25	10.0.1.5	-	INBOUND
2017-01-24T03:43:58-0400	121.48.18.25	10.0.1.5	-	INBOUND
2017-01-24T03:53:58-0400	121.48.18.25	10.0.1.5	-	INBOUND
2017-01-24T04:03:58-0400	121.48.18.25	10.0.1.5	-	INBOUND
2017-01-24T04:04:27-0400	10.0.1.10	16.180.57.224	T	OUTBOUND
2017-01-24T04:05:14-0400	10.0.1.10	200.112.224.16	T	OUTBOUND
2017-01-24T04:13:58-0400	121.48.18.25	10.0.1.5	T	INBOUND
2017-01-24T04:17:39-0400	10.0.1.10	117.215.172.107	T	OUTBOUND
2017-01-24T04:22:14-0400	10.0.1.10	84.190.173.0	T	OUTBOUND

Security Log Analysis

NSF Cybersecurity Summit 2017 -- August 15th, 2017

CTSC

Break it down (getting a sub string)



`substr(<string>, <starting index*>, <length of substring>)`
(*starting index is from 1, not 0.)

`substr("this is easy", 9, 4);`

easy

`($1 = 2017-01-24T04:03:58-0400)`

`substr($1,12,2)`

04

Break it down (Doing comparisons)



`$0!~/^#/` (Don't print lines starting with comment characters)

`$4=="T" && $5=="INBOUND"` (Successful inbound logins)

`if (true) { do something } else { do something else }`

`if (hour < 9 || hour >= 17) { print } (♫ Not Workin' 9 to 5 ♫)`

`true && true || false { print }`

Exploit: Logins vs. non-work time



(Check for inbound successful logs not between 9am and 5pm)

```
$ cat ssh.log |  
bro-cut -C -d ts id.orig_h id.resp_h auth_success direction |  
awk -F\\t '$0!~/^#/ && $4=="T" && $5 == "INBOUND"  
{ hour=int(substr($1,12,2)); if (hour < 9 || hour >= 17)  
{print}}' | less -S
```

2017-04-01T06:45:18-0400	154.19.91.90	10.0.4.26	T	INBOUND
2017-04-01T06:47:13-0400	154.19.91.90	10.0.1.5	T	INBOUND
2017-04-01T19:05:44-0400	154.19.91.90	10.0.1.5	T	INBOUND

Exploit: Logins vs. non-work time



- Alternate way using modulus of epoch time.
- % is modulus operator, gives you the remainder after division.
- Unix epoch time modulo 86400 will give you the same time of day no matter what the day

```
$ cat ssh.log |  
awk -F\\t '$8=="T" && $9 == "INBOUND" &&  
($1 % 86400 < 43200 || $1 % 86400 > 75600) {print}' |  
less -S
```

Exploit: Two factor auth analysis



Can we determine if our first level authentication has been compromised by analyzing our second level authentication?

Duo log format:

```
Timestamp,User,Integration,Factor,IP \
Address,Result,Reason,Enrollment,Device,Country,State,City
2016-09-05 16:48:22 UTC,jskeens,ACME VPN,Duo \
Push,34.207.155.20,SUCCESS,,No,,US,Texas,Houston
```

Exploit: Two factor auth analysis



(Show all failures to auth to Duo)

```
$ awk -F, ' $6=="FAILURE" ' duolog.csv
```

(Show failures from non-work geo locations)

```
$ awk -F, ' $6=="FAILURE" && $10 == "US" && $12 !~  
/(Houston|New York City)/ ' duolog.csv
```

(Or you may want to look for successes too for comparison)

Break it down: awk array primer



- An array stores a set of values. `[a, 10, "kiwi", "192.168.0.5", ...]`
- You can perform operations on the array and its values.
- `a[0] = "Tabitha Gallagher"` (store a value by numeric index)
- `a['tgallagher'] = "Tabitha Gallagher"` (store a value by text key)
- `users['ishort']['name'] = "Ira Short"` (A multi-dimensional array is an array of arrays)
- `users['ishort']['age'] = 45` (now the 'ishort' key points to an array with 2 keys)
- `users['jmoody']['name'] = "Jared Moody"`
- `for (u in users) { print users[u]['name'] }` (Loop over keys in an array)
- `length(users)` (Get the number of keys in the 'users' array)

Exploit: Two factor auth analysis (cont.)



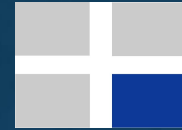
(Show failures followed by a success on the same day.)

```
$ awk -F, '{ key=substr($1,1,10) $2; if($6=="FAILURE"){  
a[key]++ }; if($6=="SUCCESS" && a[key]){ printf("%s,%s\n",  
a[key] " Failures prior", $0); a[key]=0}}' duolog.csv
```

(Show same user logs from two different geo locations on same day.)

```
$ awk -F, '/^2/ && $10!="" { key1=substr($1,1,10) $2; key2=$10  
"," $11 "," $12; a[key1][key2]=$0; if(length(a[key1]) > 1){  
for (i in a[key1]) { print a[key1][i] } } }' duolog.csv | sort  
| uniq
```

Exploit: Brute force attempts



Can I analyze login data to look for indications of someone trying to brute force my AD account?

Do you have access to authentication logs?

Yes, they are downloadable in CSV format:

```
Timestamp (EDT),IP,Type,Country  
2017-05-31 14:40:05,10.234.33.141,4771,US  
2017-05-31 14:38:27,10.234.33.141,4771,US
```


Exploit: Logins vs. non-work time



```
$ cut -c1-10 mkrenz_ad.csv | sort | uniq -c | tail -n 20
```

```
→ 286 2017-05-20
```

```
296 2017-05-21
```

```
237 2017-05-22
```

```
257 2017-05-23
```

```
248 2017-05-24
```

```
280 2017-05-25
```

```
19 2017-05-26
```

```
1 2017-05-29
```

```
861 2017-05-30
```

```
27398 2017-05-31
```

Phases of Compromise



1. Reconnaissance
2. Exploitation
3. **Reinforcement**
4. Consolidation
5. Pillage

Compromise 3: Reinforcement



How can we detect when someone installs a backdoor?

- What type of service is being backdoored?
 - SSH
- How could we tell if it's been backdoored?
 - SSH server version number change
 - Server side binary file size or checksum
- What logs can we use for version change?
 - Bro's ssh.log
- What tool can we use to detect a change?
 - awk: Store the last version seen and compare with current line's version
 - `if (lastversion != $4) { print; lastversion=$4 }`

Compromise 3: Reinforcement



From software.log:

```
Jul 27 19:32:19 141.142.227.45 22 SSH::SERVER OpenSSH_6.6.1p1
Jul 27 20:29:39 141.142.227.45 22 SSH::SERVER OpenSSH_6.6.1p1
Jul 27 22:27:53 141.142.227.45 22 SSH::SERVER OpenSSH_6.6.1p1
Jul 27 23:30:34 141.142.227.45 22 SSH::SERVER OpenSSH_6.5.1p1
```

Compromise 3: Reinforcement



From software.log:

```
Jul 27 19:32:19 141.142.227.45 22 SSH::SERVER OpenSSH_6.6.1p1
Jul 27 20:29:39 141.142.227.45 22 SSH::SERVER OpenSSH_6.6.1p1
Jul 27 22:27:53 141.142.227.45 22 SSH::SERVER OpenSSH_6.6.1p1
Jul 27 23:30:34 141.142.227.45 22 SSH::SERVER OpenSSH_6.5.1p1
```

Compromise 3: Command



```
$ cat software.log |  
bro-cut -C -d ts host host_p unparsed_version |  
awk -F\\t '$2=="141.142.227.45" && $3=="22"  
{ if (lastversion != $4) { print; lastversion=$4 } }'
```

→

```
Jul 27 22:27:53 141.142.227.45 22 OpenSSH_6.6.1p1  
Jul 27 23:30:34 141.142.227.45 22 OpenSSH_6.5.1p1
```

Finding Compromised Accounts



Bro Logs



- mysql
- http
- irc
- conn
- software

Authentication Logs



1. Successful login by "deprovisioned" user
2. Logins at strange hours (or during vacation)
3. Obvious scanning activity (ie. 2+ different users, 1 IP)
4. Multiple failures followed by a success
5. GeoIP data (where is the user coming from?)
6. Authentication from non-client device (ie. web server to web server)

MySQL Logs



- ts: Timestamp for the event
- uid: Unique ID for the connection
- id: orig_h, orig_p, resp_h, resp_p
- cmd: The command that was issued
- arg: The argument issued to the cmd
- success: Did the command succeed?
- rows: The number of affected rows
- response: Server message

MySQL Logs (mysql.log)



1. Detect vulnerable web servers (SQLi)
2. Look for large number of rows being returned.
3. `select unhex('7F454C40000E9...00')` into dumpfile
`'/usr/lib64/mysql/plugin/unknown/xiaoji64.so'`
4. Multiple authentication failures
5. Authentication from unsanctioned IPs.

MySQL log analysis



(Checking for large number of returned rows)

```
$ cat mysql.log |  
bro-cut -C -d ts id.orig_h id.resp_h success rows |  
awk -F\\t '$3=="T" && $4 > 1000 { print }'
```

(Check if any thing ran a select into dumpfile. Simple.)

```
$ grep -i "select.*into dumpfile" mysql.log
```

MySQL log analysis (cont.)

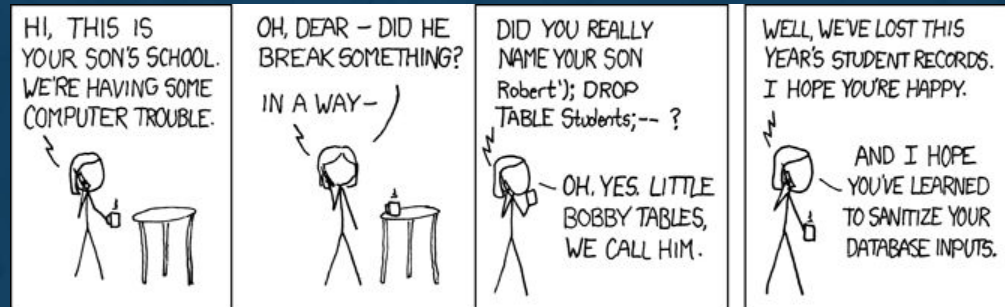


(SQL queries coming from odd networks or hosts)

```
$ cat mysql.log |  
bro-cut -C -d ts id.orig_h id.resp_h success |  
awk -F\\t '$2 !~ /^172\\.16\\.50\\. / && $3=="T" { print }'
```

(Detecting SQL exploit attempts)

```
$ grep -- "--" mysql.log
```



HTTP Logs



id.resp_h	91.134.161.42
id.resp_p	80
method	GET
host	www.ncsa.edu
uri	/
referrer	shibboleth.ncsa.edu/idp/profile/SAML2
user_agent	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
status_code	200
resp_mime_types	text/html

HTTP Logs



- Referrer mismatch
- File type mismatch (a.jpg is x-dosexec)
- Header abuse (HTTPoxy)

Phases of Compromise



1. Reconnaissance
2. Exploitation
3. Reinforcement
4. **Consolidation**
5. Pillage

IRC Logs



10.0.5.87:4766 -> 164.32.77.23:6667

JOIN #miraic

irc.log	Value
id.orig_h	10.0.5.87
id.orig_p	4766
id.resp_h	164.32.77.23
id.resp_p	6667
nick	botl656
user	botl656
command	JOIN
v	
value	#miraic

IRC Logs



- Infected host connects back to C&C channel on IRC server to check in and receive commands.

Phases of Compromise



1. Reconnaissance
2. Exploitation
3. Reinforcement
4. Consolidation
5. **Pillage**

Conn Logs



- Exfil - large flows
- Protocol mismatches (ssh over tcp/80)
- Missing protocols (udp/53, but not DNS)
- Entropy analysis

Pillage: Large exfiltrations of data



(Large outbound transfers from sensitive networks)

```
$ cat conn.log |  
bro-cut -C -d ts id.orig_h id.resp_h resp_ip_bytes |  
awk -F\\t '$3~/^172\\.17\\.50\\. / && $4 > 1000000000  
{ print }'
```



Image source: <http://en.rocketnews24.com/>

Pillage: Large exfiltrations of data



(Large outbound transfers from sensitive networks)

```
$ cat conn.log |  
bro-cut -C -d ts id.orig_h id.resp_h resp_ip_bytes |  
awk -F\\t '$3~/^172\\.17\\.50\\../ && $4 > 100000000  
{ print }'
```

→

2017-05-26T13:08:32-0400	172.17.50.7	172.17.49.42	3020603598
2017-05-26T15:11:04-0400	172.17.50.7	16.58.192.193	5031339532
2017-05-26T18:09:24-0400	172.17.50.2	57.49.32.164	171755661
2017-05-26T22:15:40-0400	172.17.50.8	172.16.9.5	1420997210

Pillage: Protocol mismatch



(Show instances of ssh running on port 80 or 443)

```
$ cat conn.log |  
bro-cut -C -d ts id.orig_h id.resp_h id.resp_p service |  
awk -F\\t '($4 == 80 || $4 == 443) && $5 == "ssh"  
→  
2017-05-02T04:03:34-0400      172.17.40.104      42.71.10.49  443  ssh
```



Putting It All Together: Intelligence



Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

IP addresses involved:

64.39.106.131 is doing recon scans

Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

URLs:

<http://18a43zad864bo96.armlay.gdn/>

is hosting malware

Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

Software versions:

OpenSSH_6.5.1p1 is outdated and might be trojaned

Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

E-mail addresses:

vladg@illinois.edu is sending malicious attachments

Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

Domain names:

[ncsa.eu](#) is a phishing site

Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

File hashes:

3aac91181c3b7eb34fb7d2b6dd673f4827fcf07

is a Flash exploit

Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

File names:

`r00tkit.tgz` is a legitimate Linux ISO

Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

SSL certificate hashes:

EC:50:1C:4A:....:A2:4C:C6:60:CC:49:03:86 is the default Ubuntu snake oil key with a well known private key

Post-Mortem Analysis



A system or account was compromised. Search *.log and generate indicators of compromise:

Pub key hashes:

a1:73:d1:e1:25:72:79:...:ed:81:bf:67:98

is a trojaned sshd

Post-Mortem Analysis



Once you have indicators of compromise, you can search your historical logs to find other compromised accounts or systems. Some might have different TTPs, which generate more indicators.

Post-Mortem Analysis



Perhaps more importantly, you can use Bro's Intel framework to load your indicators and be alerted when they're seen in the future.



Indicator	Type	Comment	Notice?
64.39.106.131	ADDR	Recon scan	F
http://18a43zad864bo96.armlay.gdn/	URL	Hosting malware	T
ncsa.eu	DOMAIN	Phishing site	T
r00tkit.tgz	FILE_NAME	Malware	T



CENTER FOR TRUSTWORTHY
SCIENTIFIC CYBERINFRASTRUCTURE
The NSF Cybersecurity Center of Excellence

CTSC webinar series: trustedci.org/webinars
Mailing list: trustedci.org/ctsc-email-lists

We thank the National Science Foundation (grant 1547272) for supporting our work.

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF.



CENTER FOR TRUSTWORTHY
SCIENTIFIC CYBERINFRASTRUCTURE
The NSF Cybersecurity Center of Excellence

NCSA:

ncsa.illinois.edu

 [@NCSAatIllinois](https://twitter.com/NCSAatIllinois)

CTSC:

trustedci.org

 [@TrustedCI](https://twitter.com/TrustedCI)

CACR:

cacr.iu.edu

 [@iucacr](https://twitter.com/iucacr)

Bro:

bro.org

 [@Bro_IDS](https://twitter.com/Bro_IDS)

SWAMP:

continuousassurance.org

 [@SWAMPTEAM](https://twitter.com/SWAMPTEAM)

CLI Magic:

climagic.org

 [@climagic](https://twitter.com/climagic)

Thank You

We thank the National Science Foundation (grant 1547272) for supporting our work.

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF.